LA-UR-02-4849

Approved for public release; distribution is unlimited.

Title:	Use of Predictive Performance Modeling During Large-Scale System Installation
Author(s):	Darren J. Kerbyson Adolfy Hoisie Harvey J. Wasserman
Submitted to:	1st Int. Workshop on Hardware/Software Support for Parallel and Distributed Scientific and Engineering Computing, SPDEC-02, Charlottesville, September 2002

Use of Predictive Performance Modeling During Large-Scale System Installation

Darren J. Kerbyson, Adolfy Hoisie, and Harvey J. Wasserman

Parallel Architectures and Performance Team
Modeling, Algorithms and Informatics Group, CCS-3
Los Alamos National Laboratory
Los Alamos, NM 87545
Email: {djk, hoisie, hjw}@lanl.gov

Abstract

In this paper we describe an important use of predictive application performance modeling – the validation of measured performance during a new large-scale system installation. Using a previously-developed and validated performance model for SAGE, a multidimensional, 3D, multi-material hydrodynamics code with adaptive mesh refinement, we were able to help guide the stabilization of the first phase of the Los Alamos ASCI Q supercomputer. We review the salient features of an analytical model for this code that has been applied to predict its performance on a large class of tera-scale parallel systems. We describe the methodology applied during system installation and upgrades to establish a baseline for the achievable "real" performance of the system. We also show the effect on overall application performance of certain key subsystems such as PCI bus speed and multirail networks. We show that utilization of predictive performance models is also a powerful system debugging tool.

1. Introduction

We have previously reported the development and validation of an analytical model that captured the performance and scaling characteristics of an important ASCI application [1]. We have also described one interesting use of the model to predict the effect on runtime of a key algorithmic change to the application enabling a different parallel decomposition method.

In this paper we report another interesting use of this same model. Los Alamos National Laboratory (LANL) is currently involved in the installation of a Tera-scale computing system called ASCI Q that comprises a large and growing number of compute servers with an interconnect fabric composed of federated switches. The installation of a system with such a large number of components is subject to a variety of both hardware- and software-related issues that effectively result in a "stabilization period" during which the system's performance may be sub-par. The question is: how does one identify sub-par performance in a large-scale parallel system, especially one that is larger than any previously available for testing. Performance observations made on a newly-installed system do not necessarily

represent just the cost of processing the workload but often include temporary idiosyncrasies of the hardware and system software, i.e., bugs, faulty or poorly configured hardware components, and so on.

We report here our experiences using a performance model to validate the measured performance during system integration of (part of) ASCI Q. Several sets of measurements of the application performance were made on the system during installation over a period of months. Only after several iterations of hardware refinements and software fixes did the performance of the system achieve the performance predicted by the model. The model did not necessarily reveal precisely what hardware/software refinements were needed; however, it was ultimately the only way to determine that no such further refinements were necessary. Along the way the model and corresponding system measurements exposed several important performance characteristics associated with ASCI Q, such as the effect of PCI bus speed and the effect of multi-rail networks on overall application performance.

2. The Alpha-Server ES45 Supercomputing System

The system considered here consists of 512 AlphaServer ES45 nodes. Each node contains four 1-GHz Alpha EV68 processors that are internally connected using two 4-GB/s memory buses to 16 GB of main memory. Each processor has an 8-MB unified level-2 cache, and a 64-KB L1 data cache. The Alpha processor has a peak performance of 2 floating point operations per cycle. Thus this first phase of the Q machine has a peak performance of 4 Tflops.

Nodes are interconnected using the Quadrics QsNet high-performance network. This network boasts high-performance communication with a typical MPI latency of 5μs and a peak throughput of 340 MB/s in one direction (detailed measured performance data are discussed in Section 4). The Quadrics network contains two components – the Elan network interface card (NIC), and the Elite switch. The Elan/Elite components are used to construct a quaternary fat-tree topology (Figure 1). A quaternary fat-tree of dimension n is composed of 4^n processing nodes and $n.4^n$-1 switches interconnected as a delta network. Each Elite switch contains an internal 16x8 full crossbar. A detailed description of the Quadrics network can be found in [4].

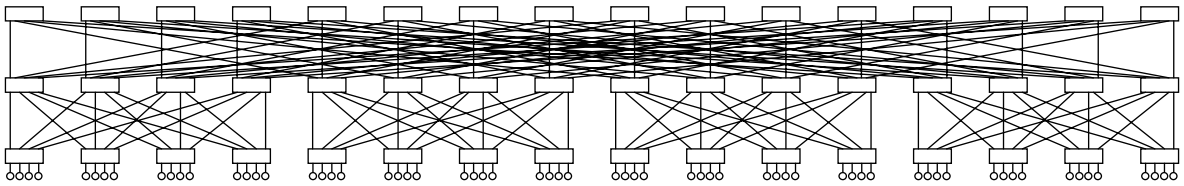

Figure 1. Network topology for a dimension 3 quaternary fat-tree network with 64 nodes

In order to implement a single rail (a single fat-tree network), a single Elan PCI interface card is used per node, in addition to a number of Elite switch boxes. The Elite switches are packaged in 128-way boxes. The first level of boxes implements the first three levels of the fat-tree and consists of 64 down and 64 up ports. The second level of boxes implements the

upper two levels of the fat-tree and consists of 128 down ports. Thus, in the 512-node system, 12 switch boxes are utilized to provide a fat-tree of dimension 5 as illustrated in Figure 2.

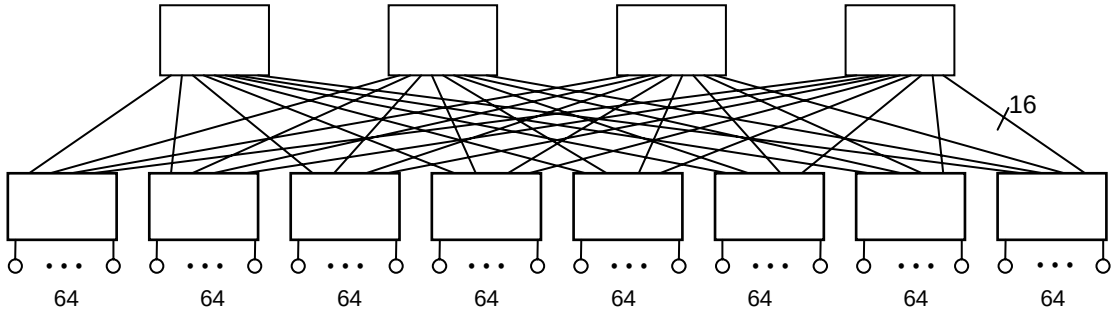

Figure 2. Interconnection of a federated Quadric network for a dimension 5 fat-tree

Using multiple independent networks, also known as "rails" is an emerging technique to overcome bandwidth limitations and to enhance fault tolerance [8]. The system being installed at Los Alamos contains two rails, i.e. two Elan cards on separate PCI interfaces per node, and two complete sets of Elite switches.

3. The Application and the Model

The application used to analyze the performance of the ES45 cluster is SAGE (SAIC's Adaptive Grid Eulerian hydrocode). It is a multidimensional (1D, 2D, and 3D), multimaterial, Eulerian hydrodynamics code with adaptive mesh refinement (AMR) consisting of 100,000+ lines of Fortran 90 code using MPI for inter-processor communications. It comes from the LANL Crestone project, whose goal is the investigation of continuous adaptive Eulerian techniques to stockpile stewardship problems. SAGE has also been applied to a variety of problems in many areas of science and engineering including: water shock, energy coupling, cratering and ground shock, stemming and containment, early time front end design, explosively generated air blast, and hydrodynamic instability problems [5]. SAGE represents a large class of production ASCI applications at Los Alamos that routinely run on 1,000's of processors for months at a time.

A detailed description of SAGE, the adaptive mesh processing, and the characteristics of its parallel scaling were described previously [1] in which we developed and validated the performance model. The salient features of the model are given in Appendix A of this paper. Table 1 gives a summary of the validation results in terms of average and maximum prediction errors across all processor configurations measured. It can be seen that the model is highly accurate with an average prediction error of 5% and maximum of 11% being typical across all machines. These data were not reported previously.

Table 1. SAGE performance model validation results

System	Number of Configurations tested	Maximum Processors tested	Maximum error (%)	Average error (%)
ASCI Blue (SGI O2K)	13	5040	12.6	4.4
ASCI Red (Intel Tflops)	13	3072	10.5	5.4
ASCI White (IBM SP3)	19	4096	11.1	5.1
Compaq AlphaServer ES40	10	464	11.6	4.7
Cray T3E	17	1450	11.9	4.1

4. Use of the SAGE model to validate system performance

The model is parametric in terms of certain basic system-related features such as the sequential processing time and the communication network performance; these had to be obtained via measurements on a small system and are listed in Table 2. The SAGE model is based on weak scaling in which the global problem size grows proportionally with the number of processors. The subgrid size remains constant at approximately 13,500 cells per subgrid.

The installation process required that the model predict performance for the first phase of the ASCI Q system with two different PCI bus speeds (initially 33 MHz and later 66MHz). The speed of the PCI bus determines the available bandwidth between the Quadrics NIC and the processor memory and thus it can have a significant impact on the performance of any parallel application. In addition, when two NICs are present within the node (in a 2-rail system), the asymptotic bandwidth increases by approximately 180% if simultaneous messages can take advantage of the two rails [4]. Individual messages are not striped across rails.

Table 2. Measured ES45 Performance Parameters for the SAGE Model

Parameter	33 MHz PCI bus	66 MHz PCI bus
$T_{comp}(E)$ (s)	0.38	0.38
$L_c(S)$ (μs)	$\begin{cases} 9.00 & S < 64 \\ 9.70 & 64 \le S \le 512 \\ 17.4 & S > 512 \end{cases}$	$\begin{cases} 6.10 & S < 64 \\ 6.44 & 64 \le S \le 512 \\ 13.8 & S > 512 \end{cases}$
$1/B_c(S)$ (ns)	$\begin{cases} 0.0 & S < 64 \\ 17.8 & 64 \le S \le 512 \\ 12.8 & S > 512 \end{cases}$	$\begin{cases} 0.0 & S < 64 \\ 12.2 & 64 \le S \le 512 \\ 8.30 & S > 512 \end{cases}$
$T_{mem}(P)$ (μs)	$\begin{cases} 1.8 & P = 2 \\ 4.8 & P > 2 \end{cases}$	$\begin{cases} 1.8 & P = 2 \\ 4.8 & P > 2 \end{cases}$

4.1 Expected Performance

The performance model was used to provide the expected performance of SAGE on the ES45 system with a 33-MHz and 66-MHz PCI bus using one and two Quadrics rails. These predictions are shown in Figure 3.

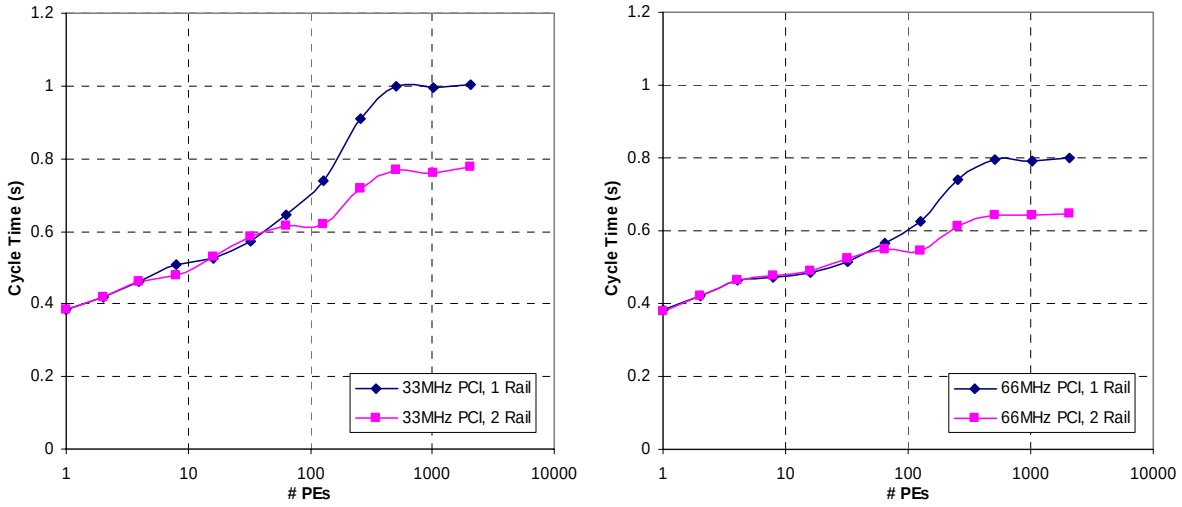

Figure 3. Performance predictions of SAGE on an ES45 system with QsNet with a) 33-MHz PCI bus, and b) 66-MHz PCI bus.

We note the following observations: 1) since the runs of SAGE were performed for weak scaling, the time should ideally be constant across all processor configurations; 2) the predicted performance is better when using 2 rails than that when using 1 rail. This occurs after a certain point (48 processors) – the point at which a domain is mapped to more than one processor. At this point a node generates more than one simultaneous out-of-node communications for gather/scatter operations whereas below this point it was only one; 3) the model predicts that the two-fold improvement in PCI bus speed results in only a 20% performance improvement in the code; 4) the SAGE cycle time is predicted to plateau above 512 processors – this is the point at which all gather/scatter communications are out-of-node.

4.2 Measured Performance

Table 3 summarizes the test conditions on each test date. The performance of SAGE was measured at several points after the installation of the machine had taken place: as soon as the machine was up and running (Sept. 9th), after a O/S upgrade from EFT 1.0 to EFT 2.0 and faulty hardware was replaced (Sept. 24th), and after an O/S patch (Oct. 24th). The upgrades that were most significant in terms of performance included bug fixes to the Quadrics RMS resource scheduling software and O/S patches that affected the priority of a process that determined the allocation of the two rails. Interestingly, this affected both 1- and 2-rail performance. These three sets of measurements, which are based on the 33-Mhz PCI bus, are compared with the model in Figure 4 (runtime vs. configuration in Figure 4a and percent error vs. configuration in Figure 4b).

The corresponding model prediction and measurements based on the 66-MHz PCI bus are shown in Figure 5. Initially (Jan 4th), not all nodes ran at 66 MHz. By Feb 2nd this had been resolved; however, not all nodes were available for testing. The Quadrics QsNet requires contiguous nodes in order to use its hardware-based collective operations. When nodes are configured out then a software component in the collectives is required which reduces overall performance. By April 20th all nodes were configured in and SAGE achieved the performance predicted by the model at all configurations except for 512 nodes.

Figures 4a and 5a show that only after all the upgrades and system debugging had taken place that the measurements matched the expected performance. Without the model, it would have been difficult to know conclusively when to stop debugging, or more importantly when *not* to. When differences did occur between the model and measurements, microkernel benchmarks were run on the computational nodes and the communication network to help identify the source of the problem.

Table 3. Summary of Test Conditions

Date	OS Version	# of Nodes in System	Performance Issues
Sept 9	EFT 1.0	128	Some faulty nodes and communication links resulted in poor / noisy communication performance, especially on 2 rails
Sept 24	EFT 2.0	128	Faulty hardware replaced but still poor 2-rail performance
Oct 24	EFT 2.1	128	2-Rail OS patch improved Quadrics Performance
Jan 4	EFT 3.0	512	PCI bus upgraded to 66 MHz; SAGE performance reverted to pre-Oct-24 performance because not all nodes successfully ran at 66 MHz.
Feb 2	EFT 3.0	512	All nodes at 66 MHz but some nodes configured out causing lower performance in collective communication such as barrier
April 20	EFT 4.0	512	All nodes configured in so barrier performance improved

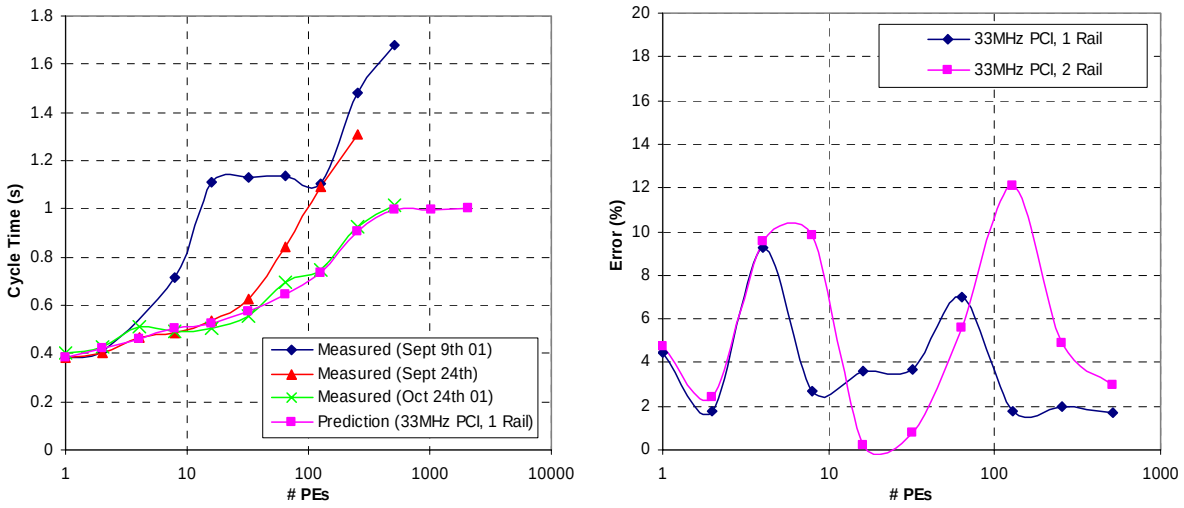

Figure 4. Measured performance of SAGE (33MHz PCI bus) compared with model predictions. a) Measurement history and model predictions using a single rail, and b) error between final measurements and model when using either 1 or 2 rails.

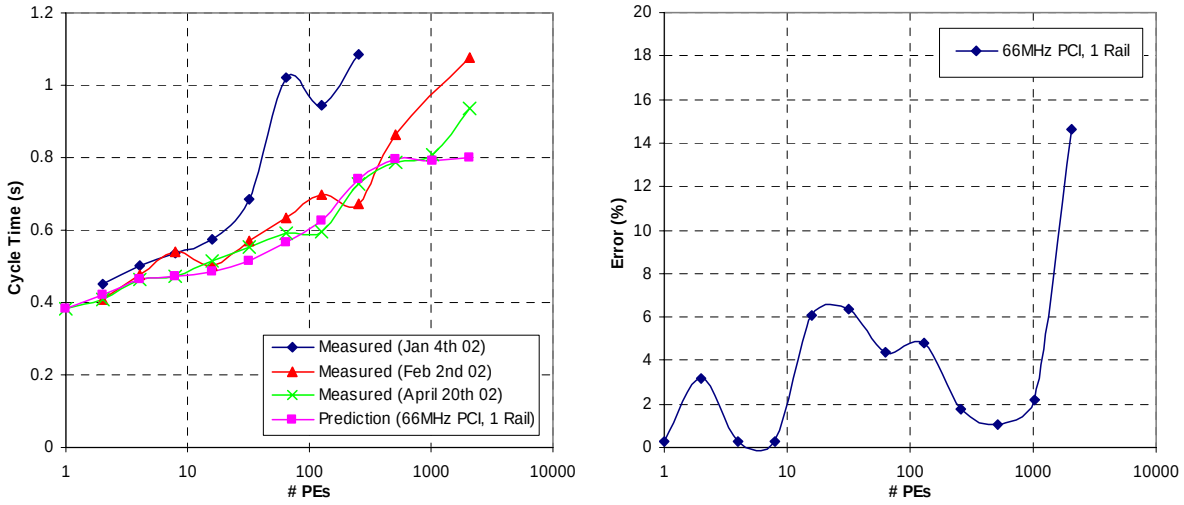

Figure 5. Measured performance of SAGE (66MHz PCI bus) compared with model predictions. a) Measurement history and model predictions using a single rail, and b) error between final measurements and model on 1 rail.

5. Summary

Our team's research over the last few years has focused on the development of analytical performance models for the ASCI workload. It has been said that modeling and predicting the performance of large-scale applications on HPC systems, is one of the great, unsolved challenges for computer science [9]. Clearly, ASCI has a critical need for information on how best to map a given application to a given architecture, and performance modeling is the

only means by which such information can be obtained quantitatively. Our approach has been successfully demonstrated for the 100,000-line+ adaptive mesh code reported here, for structured [3] and unstructured mesh transport codes [10], and for a Monte-Carlo particle transport code [11].

The work reported in this paper represents a small but important step in *applying* our performance models in a very practical way. We expect that ASCI platforms and software will be *performance engineered*, and that models will provide the means for this. The models can play a role throughout a system's lifecycle: starting at design when no hardware is available for measurement, in procurement for the comparison of systems, through to implementation / installation, and to examine the effects of updating a system over time. At each point the performance model provides an expectation of the achievable performance with a high level of fidelity. The SAGE performance model has been used for procurement purposes but company-sensitive information precludes disclosure of this in the literature. We can report, as here, how the model becomes the tool for assessing machine performance. Implementation milestone tests related to ASCI Q contractual obligations will be based partially on comparison of observed data with predictions from the SAGE model, in a manner similar to the process described in this paper.

When installing a new system, refinements to both the software system, and hardware components, are often necessary before the machine operates at the expected level of performance. The performance model for SAGE has been shown to be of great use in this process. The model has effectively provided the performance and scalability baseline for the system performance on a realistic workload. Initial system testing showed that its performance was almost 50% less than expected. After several system refinements and upgrades over a number of months, the achieved performance matched exactly the expectation provided by the model. Thus, performance models can be used to validate system performance.

References

1. Kerbyson, D.J., Alme, H.J., Hoisie, A., Petrini, F., Wasserman, H.J., Gittings, M.L.: "Predictive Performance and Scalability Modeling of a Large-scale Application," in Proc SC2001, Denver (2001)

2. Worley. P.H.: Performance "Tuning and Evaluation of a Parallel Community Climate Model," in Proc. SC99, Portland (1999)

3. Hoisie. A., Lubeck, O., Wasserman. H.: Performance and Scalability Analysis of Teraflop-Scale Parallel Architectures Using Multidimensional Wavefront Applications, Int. J. of High Performance Computing Applications, 14 (2000) 330-346

4. Petrini, F., Feng, W.C., Hoisie, A., Coll, S., Frachtenberg, E.: The Quadrics Network: High-Performance Clustering Technology, IEEE Micro, 22(1) (2002) 46-57

5. Weaver, R.: Major 3-D Parallel Simulations, BITS -Computing and communication news, Los Alamos National Laboratory, June/July, 1999, 9-11, http://www.lanl.gov/orgs/cic/cic6/bits/99june_julybits/opener.html

6. Goedecker, S., Hoisie, A.: Performance Optimization of Numerically Intensive Codes, Society for Industrial & Applied Methematics; ISBN: 0898714842 (2001)

7. Nudd, G.R., Kerbyson, D.J., et.al. PACE: A Toolset for the Performance Prediction of Parallel and Distributed Systems, Int. J. of High Performance Computing Applications, 14 (2000) 228-251

8. Coll, S., Frachtenberg, E., Petrini, F., Hoisie, A., and Gurvits, L, Using Multirail Networks in High-Performance Clusters, Proceedings of Cluster2001, Newport Beach, CA, October 2001.

9. See http://perc.nersc.gov/main.htm.

10. Kerbyson, D. J., Pautz, S. D., and Hoisie, A., "Predictive Modeling of Parallel S_N Sweeps on Unstructured Meshes", Los Alamos National Laboratory report LA-UR-02-2662, May 2002.

11. Mathis, M., Kerbyson, D. J., Hoisie, A., and Wasserman, H.J., "Performance Modeling of MCNP on Large-Scale Systems", Los Alamos Computer Science Institute Symposium, Santa Fe, October 2002.

Appendix A: SAGE model

The complete model is described below. Details on the development of the model can be found in [1].

The model assumes weak-scaling - that is the sub-domain on each processor is constant for all processor counts. In SAGE, a 3-D spatial domain is assumed which is sub-divided in 1-D only. The volume of this spatial domain is:

$$V = E.P = L^3 \tag{A.1}$$

where P is the number of PEs, and E is the number of level 0 cells per PE.

The runtime for one cycle of the code is given by:

$$\begin{aligned} T_{cycle_i}(P, E, \mathbf{D}, \mathbf{A}, \mathbf{M_{cm}}) = \; & T_{comp}(E.D_i) + T_{memcon}(P, E.D_i) + \\ & T_{allreduce}(P) + T_{GScomm}(P, E, D_i) + \\ & T_{divide}(A_i) + T_{combine}(E.D_i) + T_{load}(M_{cm_i}, P) \end{aligned} \tag{A.2}$$

where

D is the cell division factor [1..8$^{\text{maxlevel}}$], **A** is the maximum number of cells added (over all processors) through the AMR division process, and $\mathbf{M_{cm}}$ is the maximum number of cells moved between any two PEs in the load balancing. Note that **D**, **A**, and $\mathbf{M_{cm}}$ are defined as a vector whose elements are defined on a per cycle basis.

$T_{comp}(E.D_i)$	is the sequential computation time for $E.D_i$ cells (normally measured).
$T_{memcon}(P,E.\ D_i)$	is the memory contention that may occur between PEs within an SMP box
$T_{GScomm}(P,E,D_i)$	is the gather and scatter communication time
$T_{allreduce}(P)$	is the collective allreduce communication time
$T_{divide}(A_i)$	is the time to divide cells in the current cycle
$T_{combine}(E.D_i)$	the time to combine cells in the current cycle
$T_{load}(M_{cmi}, P)$	the time to perform the load-balancing

The gather-scatter communication time is given by:

$$\begin{aligned} T_{GScomm}(P, E, D_i) = C(P, E).\Big(& f_{GS\_r}.T_{comm}(Surface_Z.MPI_{Real8}, P) + \\ & f_{GS\_i}.T_{comm}(Surface_Z.MPI_{INT}, P) \Big) + \\ \Big(& f_{GS\_r}.T_{comm}(Surface_Y.MPI_{Real8}, P) + \\ & f_{GS\_i}.T_{comm}(Surface_Y.MPI_{INT}, P) + \\ & f_{GS\_r}.T_{comm}(Surface_X.MPI_{Real8}, P) + \\ & f_{GS\_i}.T_{comm}(Surface_X.MPI_{INT}, P) \Big) \end{aligned} \tag{A.3}$$

where f_{GS_r} and f_{GS_I} are the frequency of real and integer gather-scatters per cycle (measured at 160 and 17, respectively). $Surface_Z$, $Surface_Y$, $Surface_X$ are processor bounday sizes (in words) - for the 1-D slab decomposition $Surface_Z = \text{MIN}((L.D_I)^2, E.D_i/2)$, $Surface_Y = 2.L.D_i$, and $surface_x = 4.D_i$ words. MPI_{real8} and MPI_{INT} are determined by the MPI implementation

The contention on the processor network when using P processors, $C(P,E,)$ is given by:

$$C(P,E) = MIN\left(MAX\left(\frac{1}{CL} \frac{L^2}{surface_Z}, 1 \right), \frac{P_{SMP}}{CL} \right) \quad \text{(A.4)}$$

where CL is the number of communication links per node, and P_{SMP} is the number of PEs per node. $T_{comm}(S,P)$ is the time taken to communicate a message of size S when using P processors:

$$T_{comm}(S,P) = L_c(S,P) + \frac{S}{B_c(S,P)} \quad \text{(A.5)}$$

where L_c is the Latency and B_c is the Bandwidth of the communication network whose values vary with the message size and processor count.

The time taken to perform the all-reduce operations is modeled as:

$$T_{allreduce}(P) = f_{allred}.2.\log_2(P).T_{comm}(4,P) \quad \text{(A.6)}$$

where f_{allred} is the frequency of all-reduce operations per cycle. The memory contention is modeled as:

$$T_{memcon}(P,E.\ D_i) = E.D_i T_{mem}(P) \quad \text{(A.7)}$$

where $T_{mem}(P)$ is the measured memory contention on P processors per cell per cycle. The time taken to perform the cell division and cell combination at the end of each cycle is modeled as: $T_{divide}(A_i) = A_i.T_{div}$, and $T_{combine}(E.D_i) = E.D_i.\ T_{comb}$ respectively. T_{div} is the time to divide a single cell, and T_{comb} is the time to check and combine cells (both are measured on a single processor). The time taken to perform the load balancing is modeled as:

$$T_{load}(M_{cm_i}, P) = N_{vars\_i}.T_{com}(M_{cm_i}.MPI_{INT}, P) + N_{vars\_r}.T_{com}(M_{cm_i}.MPI_{Real8}, P) \quad \text{(A.8)}$$

where N_{vars_i} , and N_{vars_r} are the number of integer and real variables that are communicated in the load balance operation.